Journal of Computations & Modelling, vol.2, no.3, 2012, 1-16
ISSN: 1792-7625 (print), 1792-8850 (online)
Scienpress Ltd, 2012

Open Source Software: Quality Benefits, Evaluation Criteria and Adoption Methodologies

Lekshmy Preeti Money[1], S. Praseetha[2] and D. Mohankumar[3]

Abstract

This paper takes an in-depth look into Open Source Software (OSS), the various licensing schemes of OSS, quality benefits OSS imparts over commercial software, widely reported factors for switching to OSS and evaluation of OSS in each of those factors, methodologies to adopt OSS including a case study of InfoCache software.

Subject Classification: Software

Keywords: Open Source Software (OSS), Total Cost of Ownership (TCO), security, ease of evolution, maintainability, testability, reliability, understandability, operability, Brown field modernization, Green Field Adoption, licences, copyright

---

[1] Computer Division, VSSC,ISRO, Trivandrum, India, e-mail: lp_money@vssc.gov.in
[2] Computer Division, VSSC, ISRO, Trivandrum, India, e-mail: s_praseetha@vssc.gov.in
[3] Computer Division, VSSC, ISRO, Trivandrum, India
e-mail:d_mohankumar@vssc.gov.in

Article Info: *Received* : December 28, 2011. *Revised* : February 18, 2012
Published online : November 20, 2012

1 Introduction

Open Source Software (OSS, also known as Free and Open Source Software -FOSS) refers to software that may be freely used, modified or distributed subject to certain restrictions with respect to copyright and protection of its open source status. It must be noted that the free in FOSS doesn’t mean free of cost, but freedom to modify and/or re-distribute, [3].

The term open source software at the most basic level simply means software for which the source code is open and available. Open and available is meant to convey two concepts:

• open—The source code for the software can be read (seen) and written (modified). Further, this term is meant to promote the creation and distribution of derivative works of the software.

• available—The source code can be acquired either free of charge or for a nominal fee (e.g., media and shipping charges or online connection charges).

2 Various kinds of OSS projects

Some of the best known OSS projects are the Linux OS, the Mozilla FireFox web browser and the Apache web server. Developers commonly use OSS tools- e.g.: the scripting languages Perl, Python, PHP and Ruby; Ant and Maven for building applications; XUnit for testing; CVS and Subversion for source code control; and Eclipse or NetBeans as IDE. Open source desktop environments and related end-user desktop applications have increased both numbers and functionality.

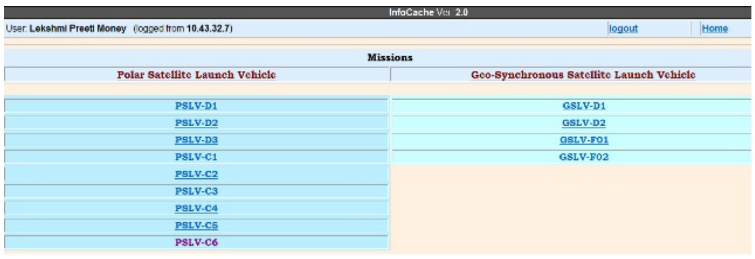
Table1: Various kinds of OSS projects

Type	**Objective**	**Control style**	**Community Structure**	**E.g.**
Exploration oriented	Sharing knowledge and innovation	Cathedral [7] style central control	Project Leader, many readers	GNU, Perl, Linux Kernel
Utility oriented	Satisfying an individual need	Baazaar [7] like decentralized control	Many peripheral developers, peer support to passive users	Linux system excluding kernel
Service oriented	Providing stable services	Council-like central control	Core project members, passive system developers	Apache, PostgreSQL

3 Infocache: A Case Study

With the increasing cost of commercial software, technology leaders are looking for ways to reduce software licensing fees. Several software systems traditionally developed using proprietary closed source software are being re-developed using open source alternatives.

To study the switching from proprietary to OSS, InfoCache- an indigenously developed software of VSSC/ISRO is considered. It is a central repository of information for storing and analyzing mission related data. This includes data related to mission performance, telemetry and Post Flight Analysis (PFA) reports. InfoCache was originally developed in Microsoft Active Server Pages (ASP) and Microsoft Access database, using Internet Information Service (IIS) web server on Windows operating system.



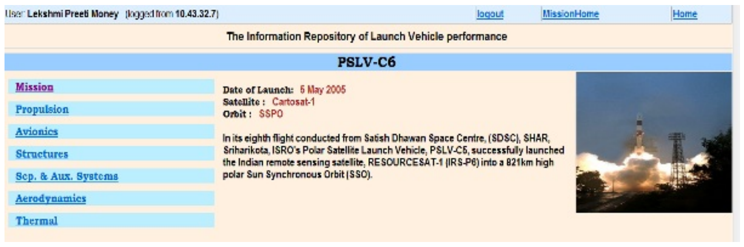
InfoCache Ver. 2.0

User: Lekshmi Preeti Money (logged from 10.43.32.7) logout Home

Missions

Polar Satellite Launch Vehicle	Geo-Synchronous Satellite Launch Vehicle
PSLV-D1	GSLV-D1
PSLV-D2	GSLV-D2
PSLV-D3	GSLV-F01
PSLV-C1	GSLV-F02
PSLV-C2	
PSLV-C3	
PSLV-C4	
PSLV-C5	
PSLV-C6	

Figure 1: Version 1.0 of InfoCache View module



User: Lekshmi Preeti Money (logged from 10.43.32.7) logout MissionHome Home

The Information Repository of Launch Vehicle performance

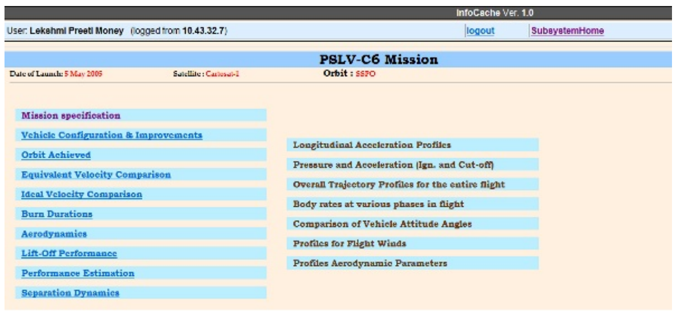
PSLV-C6

- Mission
- Propulsion
- Avionics
- Structures
- Sep. & Aux. Systems
- Aerodynamics
- Thermal

Date of Launch: 5 May 2005
Satellite : Cartosat-1
Orbit : SSPO

In its eighth flight conducted from Satish Dhawan Space Centre, (SDSC), SHAR, Sriharikota, ISRO's Polar Satellite Launch Vehicle, PSLV-C5, successfully launched the Indian remote sensing satellite, RESOURCESAT-1 (IRS-P6) into a 821km high polar Sun Synchronous Orbit (SSO).

Figure 2: Version 1.0 – Listing subsystems for selected Missions



InfoCache Ver. 1.0

User: Lekshmi Preeti Money (logged from 10.43.32.7) logout SubsystemHome

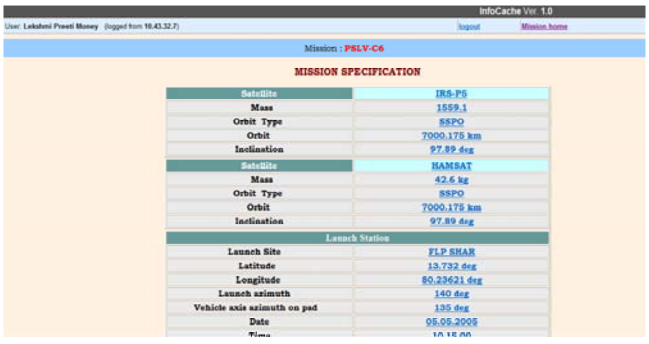
PSLV-C6 Mission

Date of Launch: 5 May 2005 Satellite : Cartosat-1 Orbit : SSPO

- Mission specification
- Vehicle Configuration & Improvements
- Orbit Achieved
- Equivalent Velocity Comparison
- Ideal Velocity Comparison
- Burn Durations
- Aerodynamics
- Lift-Off Performance
- Performance Estimation
- Separation Dynamics

- Longitudinal Acceleration Profiles
- Pressure and Acceleration (Ign. and Cut-off)
- Overall Trajectory Profiles for the entire flight
- Body rates at various phases in flight
- Comparison of Vehicle Attitude Angles
- Profiles for Flight Winds
- Profiles Aerodynamic Parameters

Figure 3: Listing pages of subsystems



InfoCache Ver. 1.0

User: Lekshmi Preeti Money (logged from 10.43.32.7) logout Mission home

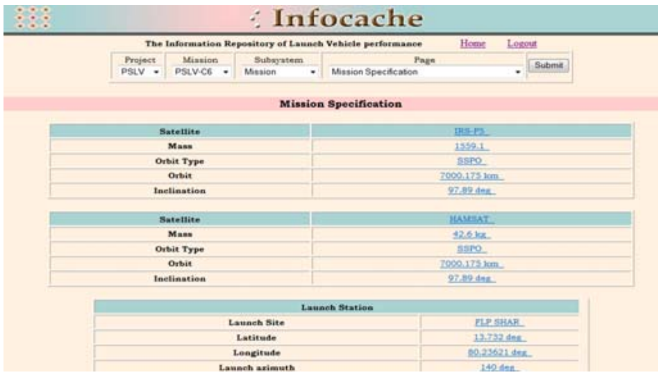
Mission : PSLV-C6

MISSION SPECIFICATION

Satellite	IRS-P5
Mass	1559.1
Orbit Type	SSPO
Orbit	7000.175 km
Inclination	97.89 deg
Satellite	HAMSAT
Mass	42.6 kg
Orbit Type	SSPO
Orbit	7000.175 km
Inclination	97.89 deg
Launch Station	
Launch Site	FLP SHAR
Latitude	13.732 deg
Longitude	80.23621 deg
Launch azimuth	140 deg
Vehicle axis azimuth on pad	135 deg
Date	05.05.2005
Time	10.15.00

Figure 4: Parameters listed for selected subsystems

Inefficiencies due to the existing software design, coupled with the organizational policy favoring development in OSS, led to the project to be re-developed using open source tools. The new project aimed at re-developing the View module (module for viewing post flight data associated with each mission) for better maintenance, more configurable display and implementation using open source solutions – Java, Java Server Pages (JSP), MySQL database and Apache Tomcat as middle tier.



Project	Mission	Subsystem	Page
PSLV	PSLV-C6	Mission	Mission Specification

Mission Specification

Satellite	IRS-P5
Mass	1559.1
Orbit Type	SSPO
Orbit	7000.175 km
Inclination	97.89 deg

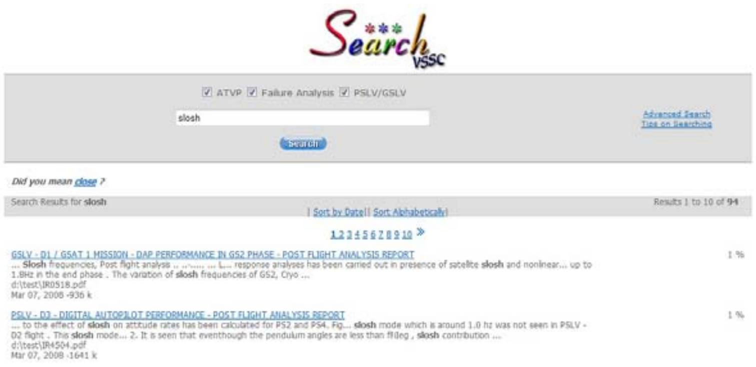
Satellite	HAMSAT
Mass	42.6 kg
Orbit Type	SSPO
Orbit	7000.175 km
Inclination	97.89 deg

Launch Station	
Launch Site	FLP SHAR
Latitude	13.732 deg
Longitude	80.23621 deg
Launch azimuth	140 deg

Figure 5: Version 2.0 InfoCache Enhanced View - Mission, Parameter selection and display in single page as opposed to hierarchical structure of pages in Version 1.0

Despite the fact that ASP supported dynamic paging, the previous module used static web pages to display data of each launch vehicle mission. Hence thousands of ASP files were maintained. Figures 1 to 4 show the various pages required to view parameters in the old version of InfoCache View module. The enhanced View module would use dynamically created web pages for display, and thus reduce the number of middle-tier files. This would eliminate manual intervention for enabling new mission data. Display properties would be easily reconfigured, since they are stored in database. Figure 5 shows the enhanced view module.

The following two functionalities would also be developed as add-ons to the existing software: - ‘Online Upload’ for mission data based on well defined workflow and online status display including e-mail notifications, ‘Advanced Post Flight Analysis Search’ with facility for full text search and highlighting search string occurrences in a document. (Ref. Figure 6). The previous Search Engine conducted searches using keywords stored in the database. The new search engine would be implemented using an open source component SearchBlox customized to SearchVSSC.



Did you mean close ?

Search Results for **slosh** Results 1 to 10 of **94**

| Sort by Date || Sort Alphabetically|

1 2 3 4 5 6 7 8 9 10 »

GSLV - D1 / GSAT 1 MISSION - DAP PERFORMANCE IN GS2 PHASE - POST FLIGHT ANALYSIS REPORT 1 %
... **Slosh** frequencies, Post flight analysis .. ..-..... ... L... response analyses has been carried out in presence of satellite **slosh** and nonlinear... up to 1.8Hz in the end phase . The variation of **slosh** frequencies of GS2, Cryo ...
d:\test\IR0518.pdf
Mar 07, 2008 -936 k

PSLV - D3 - DIGITAL AUTOPILOT PERFORMANCE - POST FLIGHT ANALYSIS REPORT 1 %
... to the effect of **slosh** on attitude rates has been calculated for PS2 and PS4. Fig... **slosh** mode which is around 1.0 hz was not seen in PSLV - D2 flight . This **slosh** mode... 2. It is seen that eventhough the pendulum angles are less than fl0deg , **slosh** contribution ...
d:\test\IR4504.pdf
Mar 07, 2008 -1641 k

Figure 6: SearchVSSC– customized from Open Source component SearchBlox

The database design would also be modified. In the earlier version of InfoCache, there were multiple MS Access databases - one for each subsystem of a particular mission. The new system envisaged a single database with multiple database tables for all missions. There would also be a shift from proprietary MS Access to an open source solution - MySQL.

InfoCache software also sought the following security enhancements:- Email ID authenticated login, Module wise access rights, Role based Access, Avoid inconsistency by record locking, Logging all transactions.

High level of security offered by OSS, platform independence, multitude of user forums, high level of community support, interoperability with other commonly available open source components and ISRO’s policy in favor of using FOSS solutions were the main reasons for choosing OSS over closed source systems in the InfoCache project.

4 Quality Benefits Imparted by OSS

OSS can enhance certain software attributes as compared to development with proprietary software. These attributes include security, ease of, evolution, maintainability, testability, reliability, understandability and interoperability, [1].

4.1 Security

Hoepman et. al. [8] argue that OSS is essential to building secure systems. Their main argument is that "opening the source allows an independent assessment of the exposure of a system, and the risk associated with using the system makes patching bugs easier and more likely and forces software developers to spend more effort on the quality of their code." They further state that OSS is easier for multiple security teams, outside vendors or independent agencies to assess. In high security organizations like ISRO, secure software – software less prone to threats-external or internal, is of paramount importance.

4.2 Ease of evolution

Another hallmark of OSS is a vibrant community; if the community is vibrant, the software is more likely to rapidly evolve new features and bug fixes. Feature innovation in OSS does not depend on a single entity – e.g.: a vendor – so the software product is easier to change and evolve. Open source also allows for code branching – the process by which a new community forms around an existing application code base to take the features and functions or underpinnings in a different direction than that of the original team. In the InfoCache scenario, the vibrant user community of Java, mySQL and Apache was a definite advantage.

4.3 Maintainability

By its nature, OSS enhances maintainability. Not only is the software more available for maintenance, but active OSS communities participate in and cultivate high level of interest in the software viability. In many cases, the code team is also part of the user population and has a vested interest in the product's functionability and growth.

4.4 Testability, Reliability, Understandability

Because OSS code is readily available, testability is easier, both functionally and structurally. Since open source communities are very comfortable with full disclosure of bugs (as opposed to hiding them or treating them as features), software users can more easily track defects and their workarounds. The argument that thorough testing is impossible because of the absence or incompetence of software requirements (as is frequently the case in OSS projects) doesn't hold because some testing approaches don't need such requirements, [9]. For the same reason that testability is easier, reliability is also higher with OSS. OSS's open nature also means understandability since there are no black boxes.

4.5 Interoperability

Interoperability is much greater with OSS because projects almost always use shared components, and these components tend to comply with relevant international standards. Frequently these standards are the main form of requirements satisfaction for all externally furnished (including open source) components available.

5 Adoption Factors and Evaluation Criteria of OSS

Here are the five distinct adoption factors and evaluation criteria for deciding whether adoption of OSS is truly beneficial to an organization. This section details the pros and cons of OSS associated with each factor, [2].

5.1 Cost Advantage

The OSS movement has always tried to downplay the fact that OSS generally doesn’t require a licence fee. But studies have shown that lower costs have largely helped drive the use of OSS. But not all OSS is free. Some products include additional service for enterprise customers, such as the certification for certain hardware, access to software updates and support services.

To estimate the costs involved in introducing OSS, an organization can calculate the Total Cost of Ownership (TCO) in the environment in which the adoption will occur. Switching costs include the costs necessary to migrate data from the old system to the new and the costs required to retrain personnel.

In the case of InfoCache, since no change in operating system was planned and the software development team being already trained in Java and mySQL, switching costs were minimal and licensing costs were greatly reduced.

5.2 Source Code Availability

The Open Source Software movement has always emphasized the advantages of source code availability. Proponents argue that making source code available lets everyone peer review the code, resulting in higher quality software. It is also suggested that it gives users more choice and control because it lets them read and modify the source code. Although many OSS advocates have proclaimed these advantages, several authors have questioned or cast doubts on them. Consider the following three scenarios

1) *Source code's availability is neither an advantage nor disadvantage:* This is because the organization never uses the source code. Organizations might have little need to modify the source code of highly mature infrastructure software such as Linux and Apache. This was the scenario in InfoCache.

2) *Organization considers the source code availability to be an advantage, but doesn't use it to study or customize the program:* Some organizations expressed a greater trust in OSS because of the source code's availability. Although they might not actually use the source code, its availability gives them the option of doing so later.

3) *OSS serves as a white box:* Organizations can use the source code to study the software's inner working or to adapt the software to their own needs. They might also customize the software e.g.: customized web mail applications.

5.3 Maturity

Another important question is whether OSS is mature enough for use in organizations aim to help decision makers decide whether a particular OSS package is mature enough to adopt. Reliability is one aspect of maturity. If an organization is unfamiliar with OSS, it should restrict its use to software that is generally considered to be mature such as Linux and Apache.

5.4 Avoiding Vendor Lock-In

Organizations frequently adopt OSS to reduce vendor lock-in and become less dependent on their software vendors. Although using OSS can reduce vendor lock-in, choosing OSS won't make an organization fully independent of vendors.

Decision makers should not adopt OSS simply to reduce their dependency on their vendor. Instead, they should investigate the degree to which the organization would depend on OSS vendors for services such as support or updates.

5.5 External support availability

Support for OSS can take different forms. Some vendors offer support contracts – e.g.: the enterprise versions of Linux distributions and the services offered by companies such as *SourceLabs*. Large software vendors such as IBM openly declare their commitment to OSS and offer various services to customers. Additionally, many independent consultancy firms will install, configure and maintain OSS systems.

Table 2 summarizes the claims and counter-claims of each of the five factors discussed above.

Table 2: Evaluation Factors for Adopting OSS

Factor	**Merits/ Claims**	**Demerits/Counter-Claims**
Cost advantage	OSS is free of charge. Linux can lower hardware costs.	Enterprise Linux isn’t free. Dual licensing (e.g. mySQL) might require a commercial license
Source Code Availability	Source code availability leads to higher quality, enables customizations, provides more choice and control and provides more trust in the software.	Lack of knowledge to apply modifications

Maturity	OSS is reliable	OSS is unreliable (claimed by advocates of proprietary software)
Vendor lock-in	OSS avoids vendor lock-in	Still dependent on OSS vendors for updates, services and support
External support	Support for OSS is available from commercial vendors	Type of support differs. Support is lacking for some OSS

6 When and how to convert to OSS

Assuming the OSS can enhance certain software qualities, an enterprise must ask two questions before deciding to use OSS to improve those qualities. Are we ready and how do we do it? The answer to the first question is complex. Readiness standards are available in Open Source Maturity Models. The answer to the second concern – how we use OSS to enhance quality assumes that the organization is culturally ready to adopt OSS and understand what areas would benefit from OSS use. To maximize quality, an organization must

- **determine the benefits and risks of OSS**:- In the case of InfoCache software, the benefits were - avoiding vendor lock-in, less cost, more secure, reliable software. The potential risk area was associated with licensing but with careful study, it was concluded that there were no copyright infringements made by using the respective OSS in InfoCache
- **determine the total costs, including ongoing and switching** : The ongoing system used static display of flight parameters maintaining several ASP files and databases. This was eliminated in the new system where four JSP files handled the business logic and the operation was database centric. Since there was no change in Operating System, the switching cost was minimized. Trained

manpower in Java and open source technologies further reduced the switching cost

- **analyze internal and external support resources:** ISRO, as an organization has always been a supporter of free and open source software. There has been a gradual shift from proprietary to OSS and many new software projects use the OSS tools and technologies. The human resources in ISRO are well trained in open source platforms. There is also good external support from the vibrant OSS community with numerous communities and forums to resolve any issues faced.
- **assess the impact of organizational performance:** The organizational performance was greatly improved with the new version of InfoCache, the enhanced view module that provided better maintenance, configurable display, better organized database and enhanced security
- **assess the health of the OSS product's ecosystem :** Since the OSS selected for InfoCache were highly mature and established ones like Apache, mySQL, Java with a broad community of developers, testers, it had high vigor and resilience.

An enterprise can begin integrating OSS in one of two ways: Green Field Adoption, which is to use OSS from the outset, or Brown Field Modernization, which is to convert existing software to OSS.

7 Ways to Integrate Open Source Software

7.1 Green Field Adoption

Here, the OSS is implemented either as a distro (short for distribution) or as a standalone application or as an integrated stack of related or integrated solutions. Distros incorporate the Linux kernel wrapped with a series of customized desktop and server maintenance applications and a load and install routine. A standalone solution can be any open source equivalent to commercial software (e.g. a CRM package), which could run either out-of-the-box or customized. Because the open source code is available, the customization could include integration with other enterprise software, as well as extensions to the internal functions of the application itself.

7.2 Brown Field Modernization

It is the process of understanding and evolving existing software assets within an architectural framework. Modernization can take many forms such as Application portfolio management, Application improvement, Platform migration, Data architecture migration, Data warehouse deployment, Reusable software assets etc. One approach to Brown field modernization is to use certain building blocks in various combinations to create target architecture. The building blocks include:

7.2.1 Refactoring: It is a process to preserve code transformation and results in modifying or cleaning up source code without changing the platform, language or external behavior. Refactoring might be the only action taken if the code is hard to maintain or change. However, refactoring might also be required to effectively wrap the application as a service and plug it into an orchestration.

7.2.2 Translating: This usually involves converting code into another language, frequently Cobol to Java. Automated translation tools are available, but without subsequent refactoring, conversion might wind up going from spaghetti Cobol to spaghetti Java. Translation can also involve different platforms or data models.

7.2.3 Wrapping: This surrounds existing applications with an interface layer to expose some or all of its functionality as a service so that it is easily re-used in building new solutions. Sometimes wrapping is possible without refactoring the existing application, in other cases much refactoring is needed.

7.2.4 Replacement: This means eventually decommissioning an application and replacing it with an entirely new solution either designed or off the shelf. Even if the solution represents a novel vision of how to run the business, the enterprise must still know something about what the old application does.

7.2.5 Orchestration: This requires integrating OSS enabled legacy applications or newly created custom or packaged applications, components or services with the help of processes implemented using business-process enactment and monitoring services.

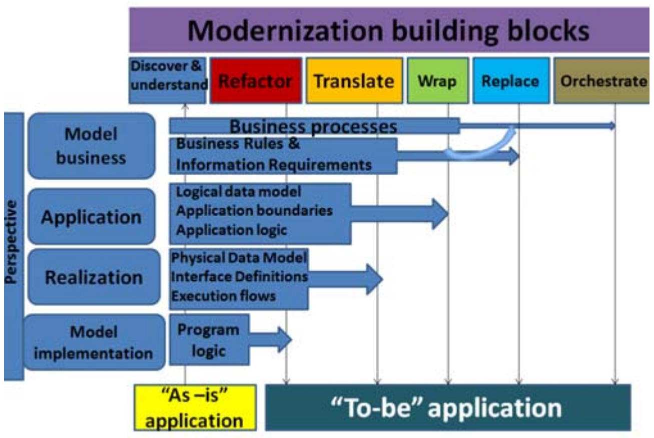


Figure7: Brown field modernization involves reengineering legacy software so that one or more qualities are significantly improved

8 Conclusion

Open Source Software is an inevitable evolution that can fulfill software engineering’s basic needs. It is increasingly important that software engineers become proficient regarding the advantages and constraints associated with specific open source components as well as legal issues related to licenses. They should not take the widely claimed advantages or disadvantages of open source software for granted, but rather should investigate how each of these claims could manifest itself in an organization-specific context.

References

[1] A.Gold, T.Costello and P. Laplante, Open Source Software:Is It Worth Converting?: *IT Professional*, (July, 2007).

[2] J. Verelst, H. Manneet, K. Ven: Should You Adopt Open Source Software : *IEEE Software*, (May, 2008).

[3] C. Ebert, Open Source Software in Industry: *IEEE Software*, (May, 2008).

[4] B.A. Calloni, J.F. McDowen and R. Stanley, Open source Software: Free Isn't Exactly Cheap! : *AIAA*, 2005-7108.

[5] S. Hissam, C.B. Weinstock, D. Plakish and J. Asund, Perspectives on Open Source Software: (November, 2001), CMU/SEI-2001-*TR-019 ESC-TR-2001-019*.

[6] http://mil-oss.org/learn-more/oss-licensing-overview

[7] E.Raymond, The Cathedral and the Bazaar http://www.catb.org/~esr/writings/cathedral-bazaar/

[8] J.-H. Hoepman and B. Jacobs, Increased Security Through Open Source: Communications of the ACM, **50**(1), (2007), 79-83.

[9] A. Elcock and P. Laplante, Testing without Requirements: Innovations in System and Software Engineering, *A NASA J.*, **2**, (December 2006), 137-145.